Question 2

RTS-GMLC grid

In [11]:
```
from vatic.engines import Simulator
from vatic.data.loaders import load_input, RtsLoader
from vatic.engines import Simulator

import pandas as pd
import numpy as np

from pathlib import Path
import dill as pickle
import os
from datetime import datetime
import matplotlib.pyplot as plt
```

In [12]:
```
import warnings
warnings.filterwarnings("ignore")
```

The following cell simulates power grid generation, with inputs **gen_data** and **load_data**

In [13]:
```
RUC_MIPGAPS = {'RTS-GMLC': 0.01}
SCED_HORIZONS = {'RTS-GMLC': 4}

grid = 'RTS-GMLC'  #For Texas, put 'Texas-7k' or 'Texas-7k_2030'
num_days = 1
start_date = '2020-01-19' #for Texas pick a date in 2018
init_state_file = None
template, gen_data, load_data = load_input(grid, start_date,
    num_days=num_days, init_state_file=init_state_file)
```

Baseline simulation:

In [14]:
```
siml = Simulator(template, gen_data, load_data, None,
                 pd.to_datetime(start_date).date(), 1, solver='gurobi',
                 solver_options={}, run_lmps=False, mipgap=RUC_MIPGAPS[grid],
                 load_shed_penalty = 1e4, reserve_shortfall_penalty = 1e3,
                 reserve_factor=0.05, output_detail=3,
                 prescient_sced_forecasts=True, ruc_prescience_hour=0,
                 ruc_execution_hour=16, ruc_every_hours=24,
                 ruc_horizon=48, sced_horizon=SCED_HORIZONS[grid],
                 lmp_shortfall_costs=False,
                 enforce_sced_shutdown_ramprate=False,
                 no_startup_shutdown_curves=False,
                 init_ruc_file=None, verbosity=0,
```

```
                output_max_decimals=4, create_plots=False,
                renew_costs=None, save_to_csv=False,
                last_conditions_file=None,)
report_dfs = siml.simulate()
```

Each simulation ouputs the following files:

In [15]:
```
report_dfs.keys()
```

Out[15]:
```
dict_keys(['hourly_summary', 'runtimes', 'total_runtime', 'ruc_summary', 't
hermal_detail', 'renew_detail', 'daily_commits', 'bus_detail', 'line_detai
l'])
```

We are interested in:

- hourly_summary: gives information about costs, over generation, renewables etc...
- thermal_detail: gives information of non-renewable generators for each hour (dispatch, headroom, unit state, unit cost)
- renew_detail: gives output for renewables generator (output -i.e. dispatch- and curtailment)
- daily_commits: gives information about which generator produces, at each hour
- bus_detail: gives information about buses (demand, mismatch, LMP (location marginal price) is not available)
- line_detail: gives information about transmission lines' flow

You are now asked to change the initial inputs, and analyse how the results change.

For the following questions, **plot the graphs** and comment them according to the question.

Question 2.1
Changes in actual demand (load data)
(a) What happens if the actual demand increases by 10%?
(b) What if it decreases by 10%?

Question 2.2
Changes in generators
(a) What happens if all RTPV generators are shut down?
(b) What happens if all HYDRO generators are shut down?

Now, you can proceed to manipulate the generators inputs!

In [ ]:
```
report_dfs
```

In [ ]:
```
load_data
```

Question 2.1

In [16]:
```
import copy
import pandas as pd

# scale only one top-level 'actl' block
def scale_load_block(load_df: pd.DataFrame, factor: float, block: str = "act
    ld = load_df.copy(deep=True)
    if isinstance(ld.columns, pd.MultiIndex):
        idx = pd.IndexSlice
        ld.loc[:, idx[block, :]] = ld.loc[:, idx[block, :]] * factor
    else:
        ld = ld * factor
    return ld
```

In [17]:
```
print("\nRunning Simulation: Demand +10% on ACTL only ...")
load_data_up = scale_load_block(load_data, 1.10, block="actl")
```

```
Running Simulation: Demand +10% on ACTL only ...
```

In [19]:
```
sim_demand_up = Simulator(
    template, gen_data, load_data_up, None,
    pd.to_datetime(start_date).date(), 1, solver='gurobi',
    solver_options={}, run_lmps=False, mipgap=RUC_MIPGAPS[grid],
    load_shed_penalty=1e4, reserve_shortfall_penalty=1e3,
    reserve_factor=0.05, output_detail=3,
    prescient_sced_forecasts=True, ruc_prescience_hour=0,
    ruc_execution_hour=16, ruc_every_hours=24,
    ruc_horizon=48, sced_horizon=SCED_HORIZONS[grid],
    lmp_shortfall_costs=False,
    enforce_sced_shutdown_ramprate=False,
    no_startup_shutdown_curves=False,
    init_ruc_file=None, verbosity=0,
    output_max_decimals=4, create_plots=False,
    renew_costs=None, save_to_csv=False,
    last_conditions_file=None,
)
report_demand_up = sim_demand_up.simulate()
print("Demand +10% (ACTL only) simulation complete.")
```

```
Demand +10% (ACTL only) simulation complete.
```

In [8]:
```

```

```
Running Simulation: Demand +10%...
Demand +10% simulation complete.
```

In [20]:
```
# Q2.1(b) - Decrease Demand by 10%
load_data_down = scale_load_block(load_data, 0.90, block="actl")

# Run the simulation with the new load data
sim_demand_down = Simulator(
    template, gen_data, load_data_down, None,
    pd.to_datetime(start_date).date(), 1, solver='gurobi',
    solver_options={}, run_lmps=False, mipgap=RUC_MIPGAPS[grid],
    load_shed_penalty = 1e4, reserve_shortfall_penalty = 1e3,
    reserve_factor=0.05, output_detail=3,
    prescient_sced_forecasts=True, ruc_prescience_hour=0,
    ruc_execution_hour=16, ruc_every_hours=24,
    ruc_horizon=48, sced_horizon=SCED_HORIZONS[grid],
    lmp_shortfall_costs=False,
    enforce_sced_shutdown_ramprate=False,
    no_startup_shutdown_curves=False,
    init_ruc_file=None, verbosity=0,
    output_max_decimals=4, create_plots=False,
    renew_costs=None, save_to_csv=False,
    last_conditions_file=None,
    )
report_demand_down = sim_demand_down.simulate()
print("Demand -10% simulation complete.")
```

```
Demand -10% simulation complete.
```

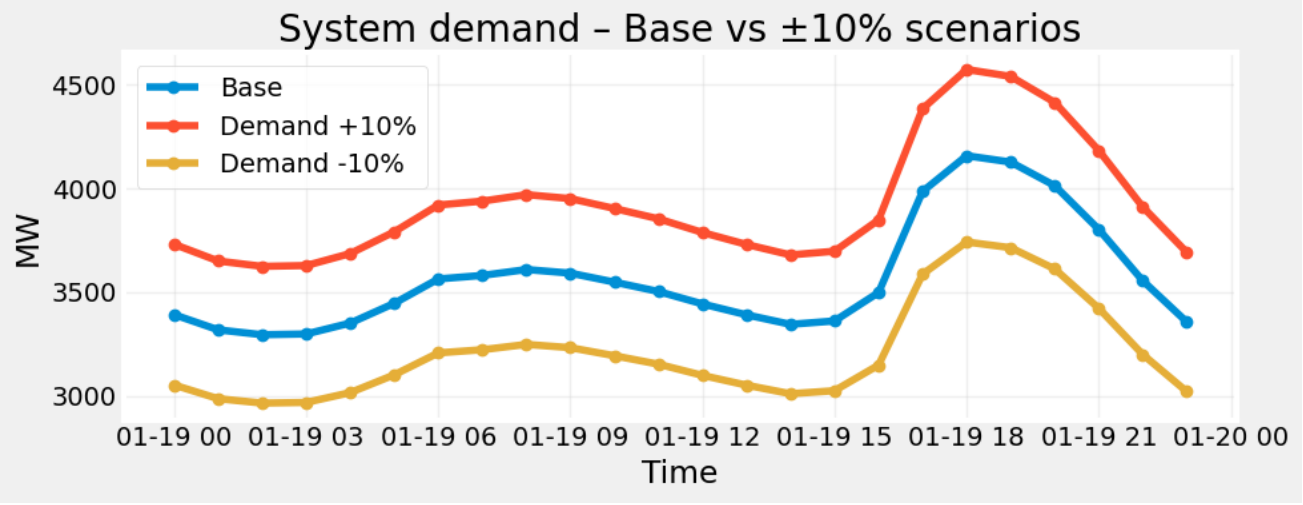
In [ ]:
```

```

In [22]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

scenarios_raw = {
    "Base": report_dfs,
    "Demand +10%": report_demand_up,
    "Demand -10%": report_demand_down,
}

def prepare_hourly(rep):

    hs = rep["hourly_summary"].copy()

    hs = hs.reset_index()

    # timestamp
    hs["Datetime"] = pd.to_datetime(hs["Date"]) + pd.to_timedelta(hs["Hour"]

    # Total cost per hour
    hs["TotalCost"] = hs["FixedCosts"] + hs["VariableCosts"]
```

```
    # Curtailment as percentage of available renewables
    hs["CurtailmentPct"] = (
        hs["RenewablesCurtailment"] / hs["RenewablesAvailable"]
    ) * 100.0
    hs["CurtailmentPct"] = hs["CurtailmentPct"].replace([np.inf, -np.inf], r

    # Datetime as index
    hs = hs.set_index("Datetime").sort_index()
    return hs

scenario_hs = {name: prepare_hourly(rep) for name, rep in scenarios_raw.item
```

In [23]:
```
def plot_metric(metric, ylabel, title, scenarios=scenario_hs):
    plt.figure(figsize=(10, 4))
    for name, hs in scenarios.items():
        if metric not in hs.columns:
            continue
        plt.plot(hs.index, hs[metric], marker="o", label=name)
    plt.xlabel("Time")
    plt.ylabel(ylabel)
    plt.title(title)
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.tight_layout()
```

In [24]:
```
plot_metric(
    metric="Demand",
    ylabel="MW",
    title="System demand — Base vs ±10% scenarios",
)
```

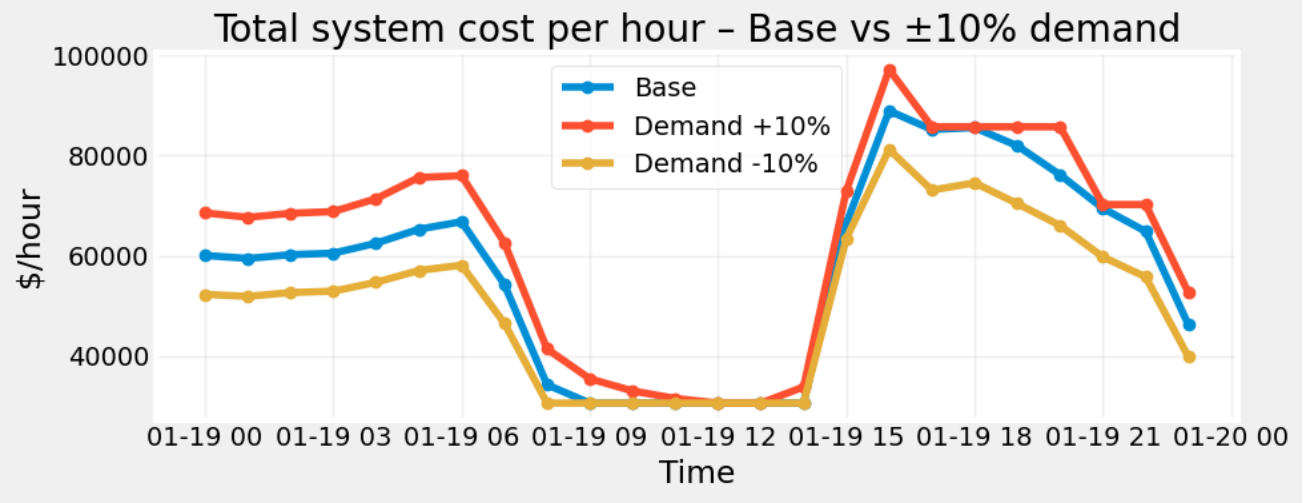


Here, the 24-hour system-load profile is shown for the baseline and 10% demand shock scenarios. Increasing demand by 10% uniformly shifts the entire load curve upward, leading to a higher evening peak around 4.5 GW. Conversely, decreasing demand by 10%

produces a proportional downward shift. The overall shape of the daily load profile remains unchanged, indicating that the timing of the morning and evening peaks is unaffected, and only the magnitude of consumption varies.

In [ ]:

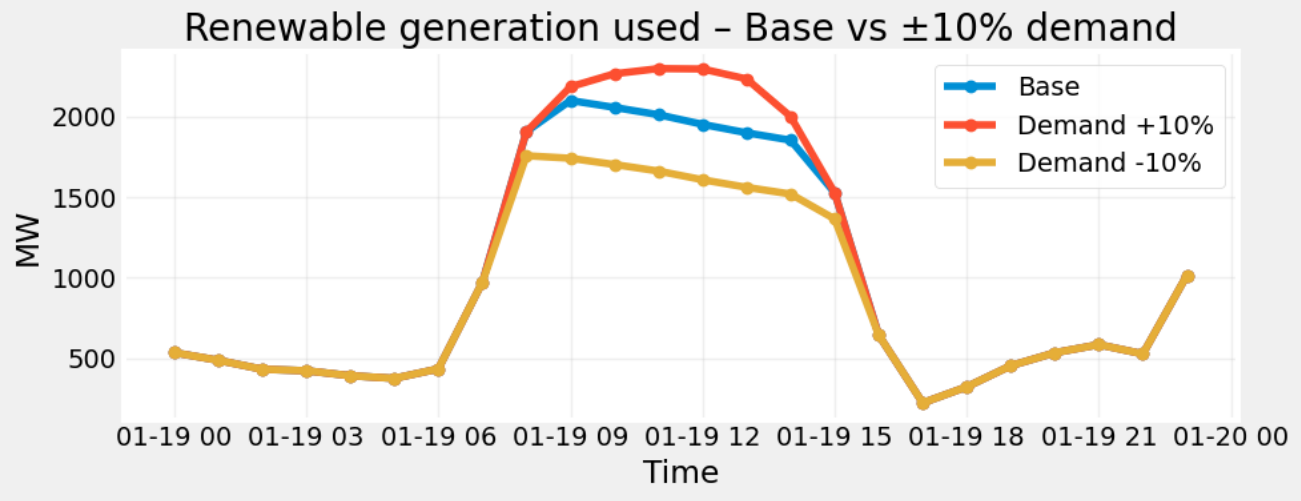
In [25]:
```
plot_metric(
    metric="TotalCost",
    ylabel="$/hour",
    title="Total system cost per hour — Base vs ±10% demand",
)
```



This figure compares total hourly system costs under 10% demand shock scenarios relative to the baseline. Increasing demand by 10% shifts the cost curve upward, with the largest rise observed during the evening ramp when higher-cost peaking generators are dispatched. Conversely, reducing demand by 10% lowers overall system cost but with a smaller magnitude of savings, since much of the reduced load can still be met by low-marginal-cost baseload units. Overall, total system cost scales non-linearly with demand, reflecting the stepped marginal-cost structure of generator dispatch.

In [ ]:

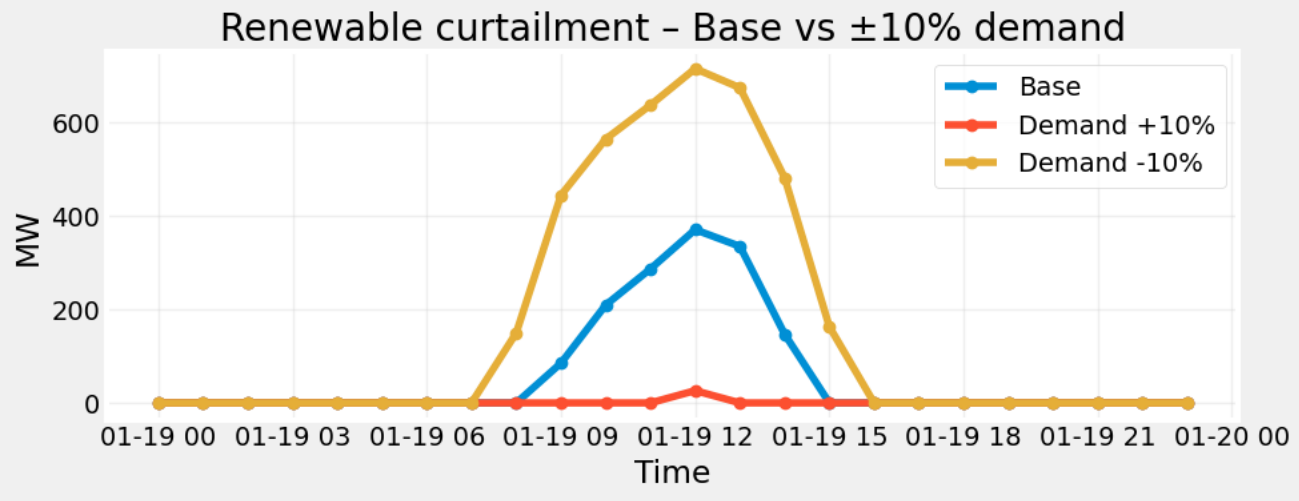
In [26]:
```
plot_metric(
    metric="RenewablesUsed",
    ylabel="MW",
    title="Renewable generation used — Base vs ±10% demand",
)
```

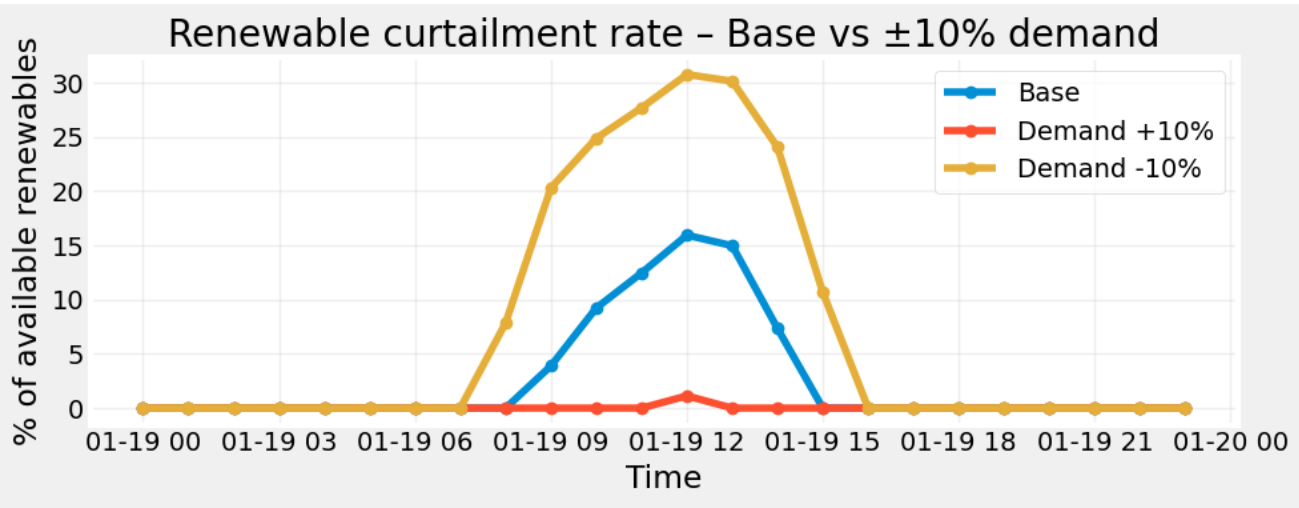


This figure shows renewable generation actually used under the baseline and 10% demand shock scenarios. The profile follows the typical solar–wind pattern—ramping up in the morning, peaking around midday, and tapering off in the evening. Higher demand enables slightly greater renewable utilization (red line above base), while lower demand increases curtailment (yellow line below base). The nearly identical shape across all cases indicates that renewable output is primarily supply-driven, with demand changes mainly affecting the extent of curtailment rather than total availability

In [ ]:

In [27]:
```
plot_metric(
    metric="RenewablesCurtailment",
    ylabel="MW",
    title="Renewable curtailment — Base vs ±10% demand",
)

plot_metric(
    metric="CurtailmentPct",
    ylabel="% of available renewables",
    title="Renewable curtailment rate — Base vs ±10% demand",
)
```



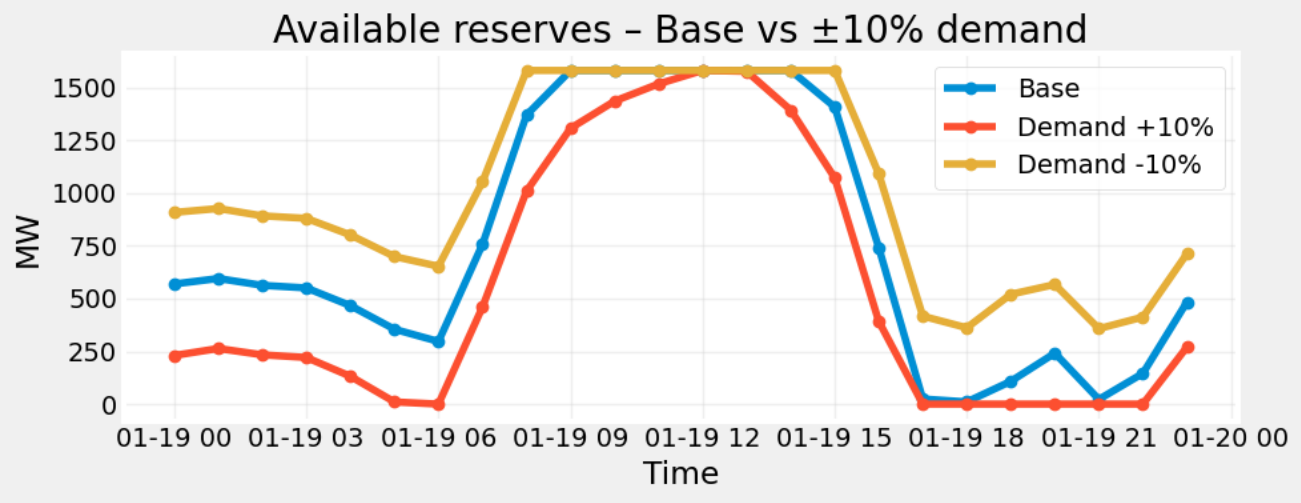


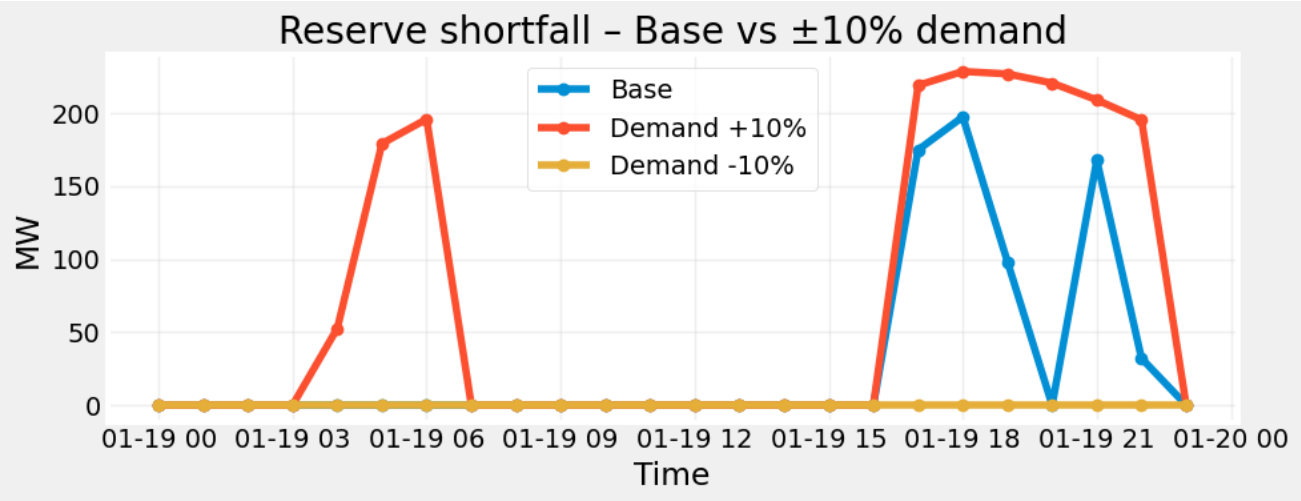
Together, these figures illustrate how renewable curtailment responds to changes in system demand. Curtailment rises sharply during midday hours when solar output is high and demand is insufficient to absorb the available renewable generation. Under the –10% demand scenario, excess renewable supply leads to significant curtailment, reaching roughly 700 MW or 25–30% of available renewables—whereas a 10% demand increase nearly eliminates curtailment, as the higher load utilizes more of the renewable output. The consistent timing of curtailment across scenarios indicates that curtailment is primarily demand-driven rather than supply-driven, arising mainly during low-load periods. These results underscore the importance of system flexibility and transmission capability in enabling full renewable integration, especially when demand dips below renewable availability.

In [ ]:

In [28]:
```
plot_metric(
    metric="AvailableReserves",
    ylabel="MW",
```

```
    title="Available reserves — Base vs ±10% demand",
)

plot_metric(
    metric="ReserveShortfall",
    ylabel="MW",
    title="Reserve shortfall — Base vs ±10% demand",
)
```





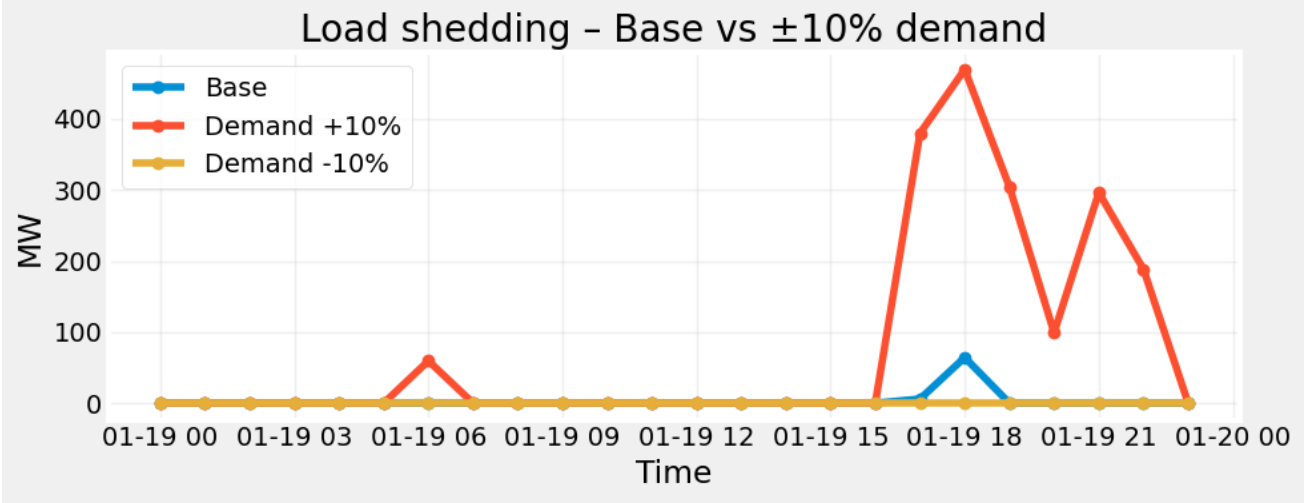
In [ ]:

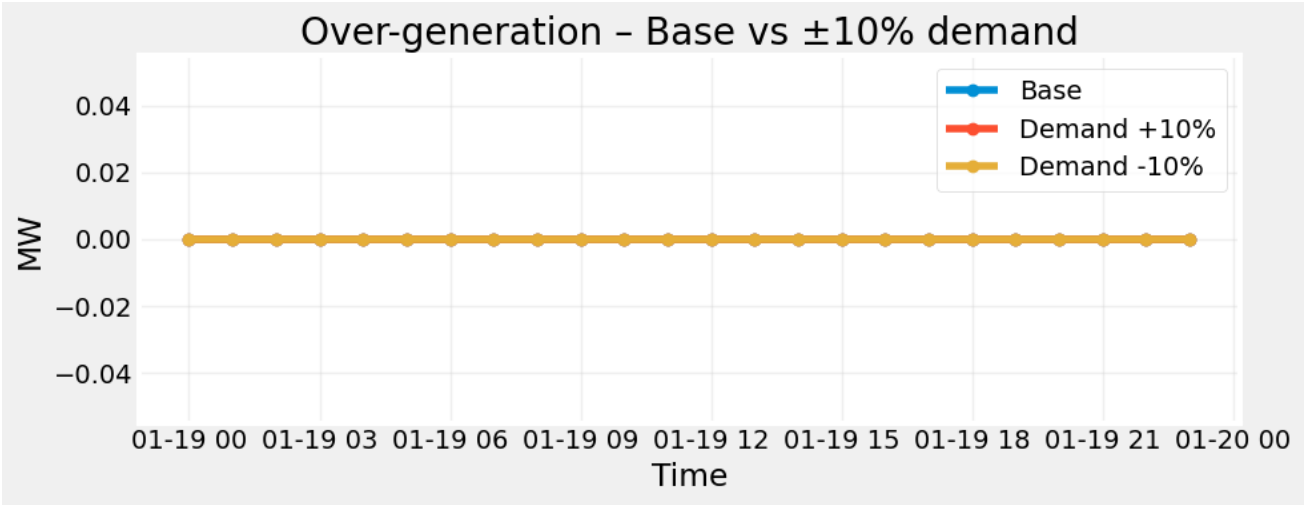
These figures show how available reserves and reserve shortfalls respond to changes in system demand. During midday, reserve margins are ample across all scenarios due to high renewable availability and moderate load. As the evening peak approaches and solar generation declines, reserves tighten sharply, and shortfalls emerge. Under the +10% demand scenario, the system operates with higher reserve commitments on average but faces the most pronounced evening shortfalls, reflecting increased operational stress. Conversely, with –10% demand, reserves remain comfortably above requirements throughout the day, indicating reduced system strain. Overall, these results

highlight that rising demand compresses reliability margins, particularly during the renewable ramp-down period when dispatchable capacity is most constrained.

In [ ]:

In [29]:
```
plot_metric(
    metric="LoadShedding",
    ylabel="MW",
    title="Load shedding – Base vs ±10% demand",
)

plot_metric(
    metric="OverGeneration",
    ylabel="MW",
    title="Over-generation – Base vs ±10% demand",
)
```



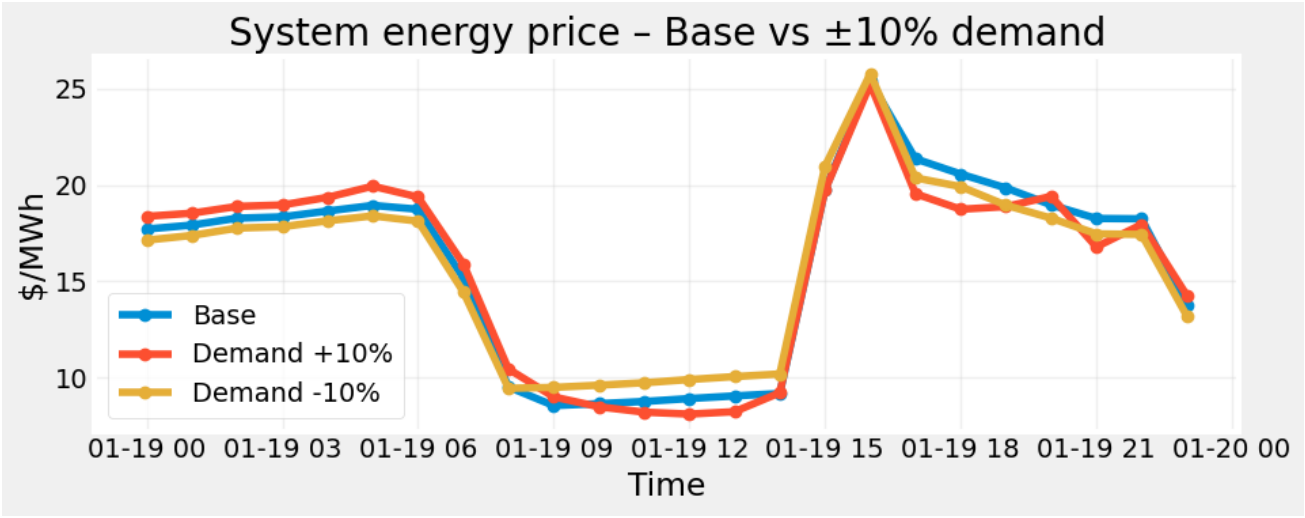


These figures illustrate the system's performance under extreme operating conditions. Load shedding occurs only during the evening peak, when renewable output declines

and reserves are exhausted. The +10% demand scenario experiences the highest unserved load (up to ~450 MW), indicating heightened system stress under heavy load. The base case shows only minimal shedding, while the –10% case remains fully reliable throughout the day. Over-generation, meanwhile, remains effectively zero across all cases, implying that curtailment controls successfully mitigated renewable surplus. Together, these results show that increasing demand sharply tightens system reliability margins, while lower demand enhances operational security but at the cost of higher renewable curtailment earlier observed.

In [ ]:

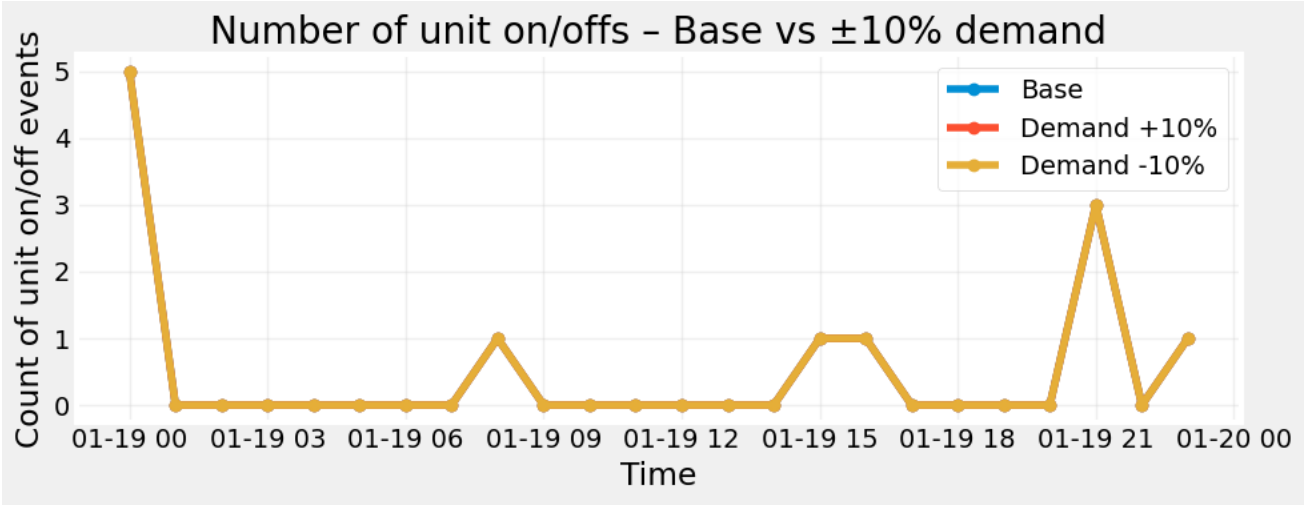
In [30]:
```
plot_metric(
    metric="Price",
    ylabel="$/MWh",
    title="System energy price – Base vs ±10% demand",
)
```

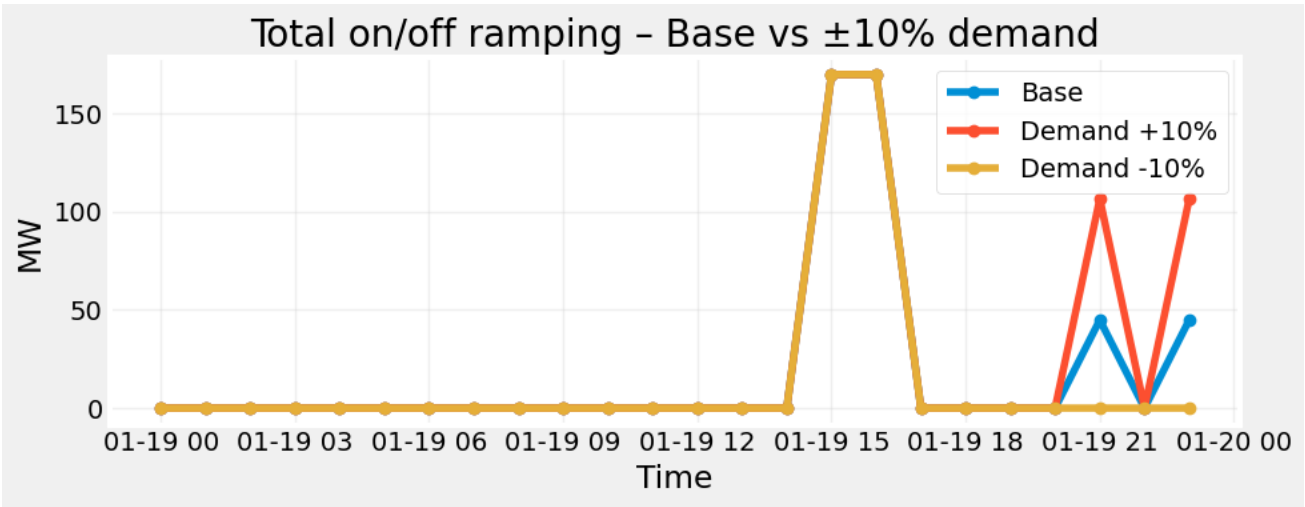


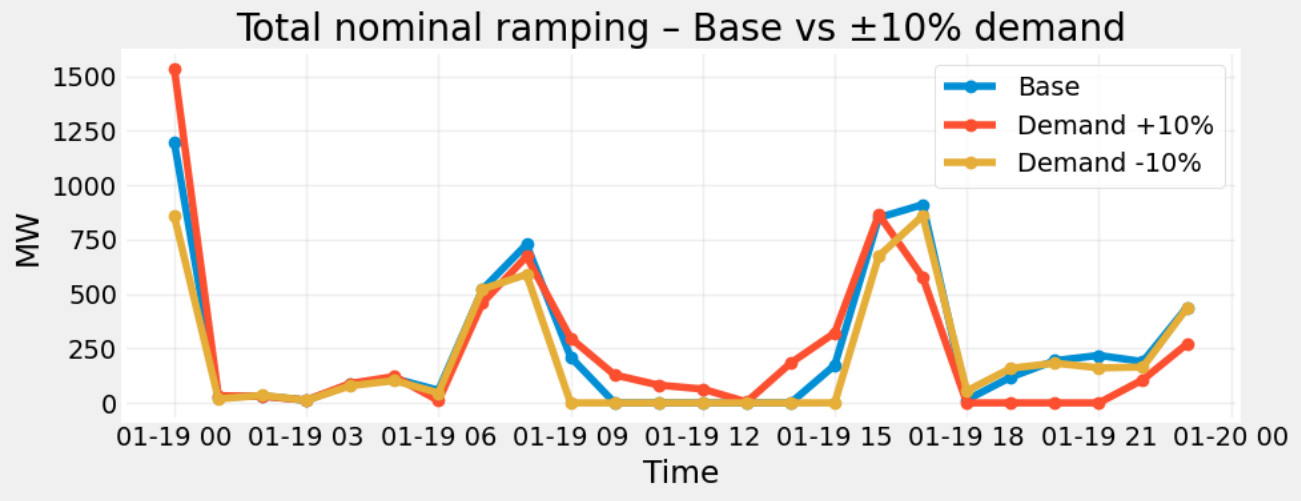
This figure presents the hourly system marginal energy price under baseline, +10%, and –10% demand conditions. Prices remain stable overnight, dip during solar-rich midday hours, and surge sharply during the evening as renewable output falls and reserves tighten. The +10% demand scenario exhibits the highest price volatility, peaking near $27/MWh due to scarcity conditions coinciding with load shedding. Conversely, the –10% demand case shows flatter, lower price levels owing to greater renewable availability and increased system slack. Overall, system energy prices closely mirror the real-time balance between renewable generation, thermal flexibility, and reserve adequacy.

In [ ]:

In [31]:
```
plot_metric(
    metric="Number on/offs",
    ylabel="Count of unit on/off events",
    title="Number of unit on/offs – Base vs ±10% demand",
)

plot_metric(
    metric="Sum on/off ramps",
    ylabel="MW",
    title="Total on/off ramping – Base vs ±10% demand",
)

plot_metric(
    metric="Sum nominal ramps",
    ylabel="MW",
    title="Total nominal ramping – Base vs ±10% demand",
)
```







The count of commitment changes is modest across all cases, with small clusters during morning start-ups and the evening ramp.

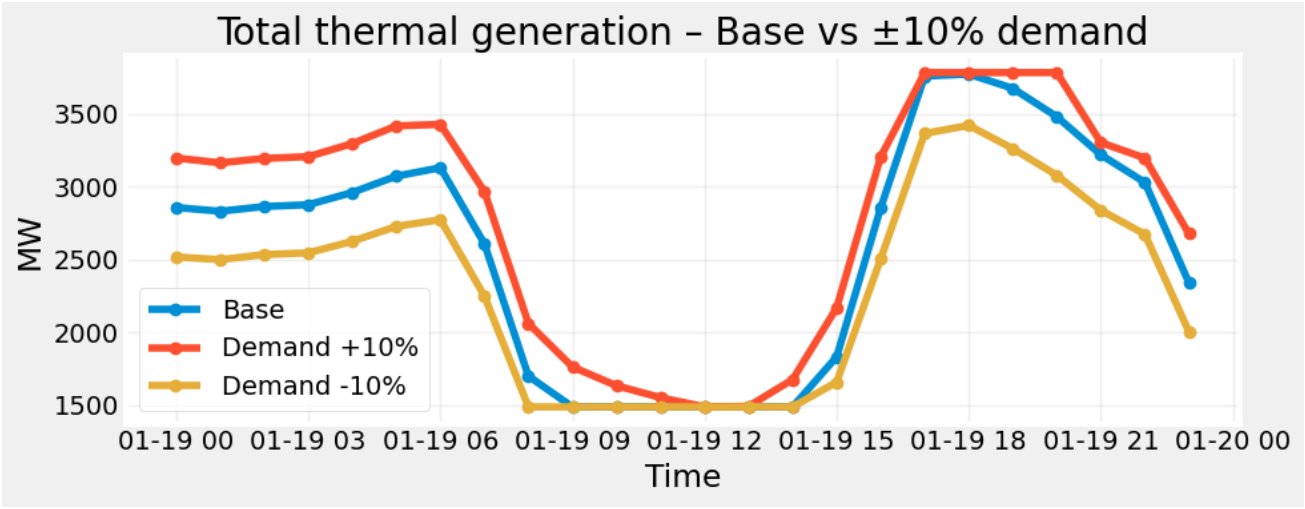
Ramping magnitudes increase materially in the +10% case during the night as units start/stop at higher outputs to meet steeper ramps. The –10% case exhibits the smallest on/off ramping because fewer large units need to be cycled.

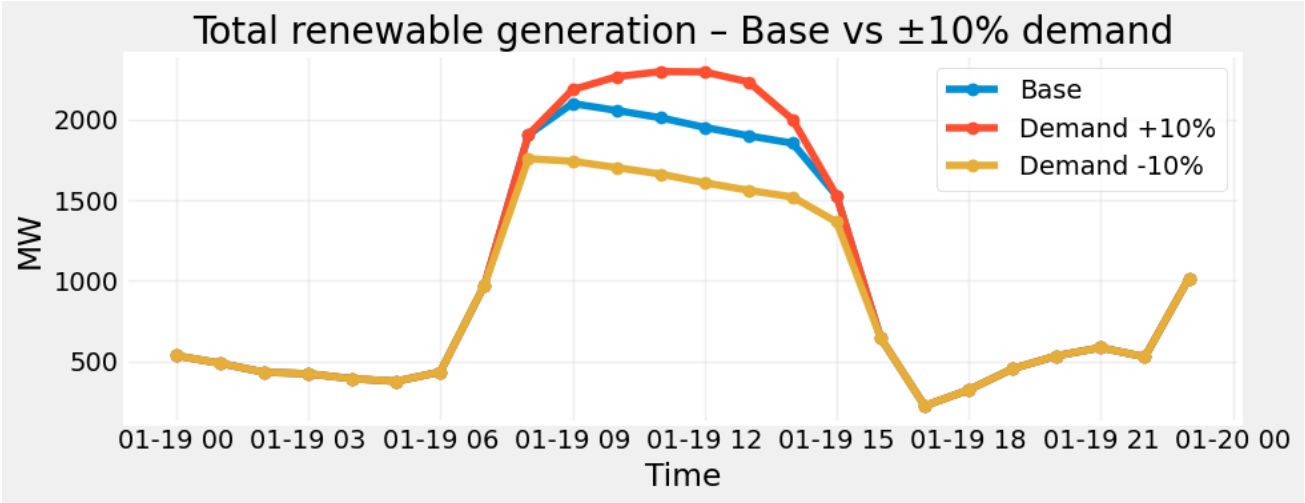
Nominal (within-state) ramping traces the solar duck-curve with a morning ramp-up, a midday lull, and a sharp evening ramp-down. Relative to Base, +10% amplifies the afternoon nominal ramp, while –10% dampens it which is consistent with higher/lower net-load gradients.

In [ ]:

In [32]:
```
def thermal_dispatch_time_series(rep):
    td = rep["thermal_detail"].copy()
    td = td.reset_index()  # make Date / Hour columns if they were index
    td["Datetime"] = pd.to_datetime(td["Date"]) + pd.to_timedelta(td["Hour"]
    return td.groupby("Datetime")["Dispatch"].sum().sort_index()

def renewable_dispatch_time_series(rep):
    rd = rep["renew_detail"].copy()
    rd = rd.reset_index()
    rd["Datetime"] = pd.to_datetime(rd["Date"]) + pd.to_timedelta(rd["Hour"]
    return rd.groupby("Datetime")["Output"].sum().sort_index()

thermal_ts = {
    name: thermal_dispatch_time_series(rep) for name, rep in scenarios_raw.i
}
renew_ts = {
    name: renewable_dispatch_time_series(rep) for name, rep in scenarios_raw
}
```

In [33]:
```
# Plot thermal dispatch
plt.figure(figsize=(10, 4))
for name, series in thermal_ts.items():
    plt.plot(series.index, series.values, marker="o", label=name)
plt.xlabel("Time")
plt.ylabel("MW")
plt.title("Total thermal generation – Base vs ±10% demand")
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()

# Plot renewable dispatch
plt.figure(figsize=(10, 4))
for name, series in renew_ts.items():
    plt.plot(series.index, series.values, marker="o", label=name)
plt.xlabel("Time")
plt.ylabel("MW")
plt.title("Total renewable generation – Base vs ±10% demand")
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
```





Thermal generation dominates during nighttime and evening hours, while renewables peak during daylight, producing a characteristic duck-curve pattern. Under the +10% demand scenario, the system relies more heavily on thermal generation, particularly during the evening peak—leading to higher operating costs and emissions. In contrast, the –10% demand case reduces thermal dispatch but increases renewable curtailment due to limited midday load. These dynamics highlight the complementary relationship between thermal and renewable resources: thermal units ensure reliability during renewable shortfalls, while renewables reduce fossil generation and system costs when available.

In [ ]:

Overall:

The system's response to 10% demand shock variations highlights the fundamental trade-offs between cost, reliability, flexibility, and renewable utilization. Higher demand (+10%) consistently increases total system cost, thermal generation, and market prices, while intensifying reserve shortfalls, ramping stress, and load-shedding during evening peaks—signs of a grid operating close to its capacity. In contrast, lower demand (–10%) reduces system cost and enhances reliability margins but leads to greater renewable curtailment due to insufficient load to absorb available clean generation. The base scenario achieves a balanced outcome with stable prices, moderate renewable use, and limited stress on reserves or generation assets.

In [ ]:

Question 2.2

In [34]:
```
(gen_data.columns)
```

Out[34]:
```
MultiIndex([('actl',    '101_PV_1'),
            ('actl',    '101_PV_2'),
            ('actl',    '101_PV_3'),
            ('actl',    '101_PV_4'),
            ('actl',    '102_PV_1'),
            ('actl',    '102_PV_2'),
            ('actl',    '103_PV_1'),
            ('actl',    '104_PV_1'),
            ('actl',    '113_PV_1'),
            ('actl',  '118_RTPV_1'),
            ...
            ('fcst',  '320_RTPV_4'),
            ('fcst',  '320_RTPV_5'),
            ('fcst',  '320_RTPV_6'),
            ('fcst', '322_HYDRO_1'),
            ('fcst', '322_HYDRO_2'),
            ('fcst', '322_HYDRO_3'),
            ('fcst', '322_HYDRO_4'),
            ('fcst',    '324_PV_1'),
            ('fcst',    '324_PV_2'),
            ('fcst',    '324_PV_3')],
           length=160)
```

In [35]:
```python
import pandas as pd

def shut_down_units(gen_df, keyword, label="", level0_target="actl"):

    df = gen_df.copy()

    lvl0 = df.columns.get_level_values(0)
    lvl1 = df.columns.get_level_values(1).astype(str)

    mask = (lvl0 == level0_target) & lvl1.str.contains(keyword, case=False,

    cols_to_zero = df.columns[mask]

    print(f"{label}: shutting down {level0_target} generators matching '{key
    print(list(cols_to_zero))

    df.loc[:, cols_to_zero] = 0.0
    return df
```

In [36]:
```python
# Scenario: all RTPV generators shut down
gen_data_rtpv_off = shut_down_units(gen_data, "RTPV", label="No RTPV")

# Scenario: all HYDRO generators shut down
gen_data_hydro_off = shut_down_units(gen_data, "HYDRO", label="No HYDRO")
```

```
No RTPV: shutting down actl generators matching 'RTPV':
[('actl', '118_RTPV_1'), ('actl', '118_RTPV_10'), ('actl', '118_RTPV_2'), ('
actl', '118_RTPV_3'), ('actl', '118_RTPV_4'), ('actl', '118_RTPV_5'), ('act
l', '118_RTPV_6'), ('actl', '118_RTPV_7'), ('actl', '118_RTPV_8'), ('actl',
'118_RTPV_9'), ('actl', '213_RTPV_1'), ('actl', '308_RTPV_1'), ('actl', '313
_RTPV_1'), ('actl', '313_RTPV_10'), ('actl', '313_RTPV_11'), ('actl', '313_R
TPV_12'), ('actl', '313_RTPV_13'), ('actl', '313_RTPV_2'), ('actl', '313_RTP
V_3'), ('actl', '313_RTPV_4'), ('actl', '313_RTPV_5'), ('actl', '313_RTPV_
6'), ('actl', '313_RTPV_7'), ('actl', '313_RTPV_8'), ('actl', '313_RTPV_9'),
('actl', '320_RTPV_1'), ('actl', '320_RTPV_2'), ('actl', '320_RTPV_3'), ('ac
tl', '320_RTPV_4'), ('actl', '320_RTPV_5'), ('actl', '320_RTPV_6')]
No HYDRO: shutting down actl generators matching 'HYDRO':
[('actl', '122_HYDRO_1'), ('actl', '122_HYDRO_2'), ('actl', '122_HYDRO_3'),
('actl', '122_HYDRO_4'), ('actl', '122_HYDRO_5'), ('actl', '122_HYDRO_6'),
('actl', '201_HYDRO_4'), ('actl', '215_HYDRO_1'), ('actl', '215_HYDRO_2'),
('actl', '215_HYDRO_3'), ('actl', '222_HYDRO_1'), ('actl', '222_HYDRO_2'),
('actl', '222_HYDRO_3'), ('actl', '222_HYDRO_4'), ('actl', '222_HYDRO_5'),
('actl', '222_HYDRO_6'), ('actl', '322_HYDRO_1'), ('actl', '322_HYDRO_2'),
('actl', '322_HYDRO_3'), ('actl', '322_HYDRO_4')]
```

In [38]:

```python
def run_sim_with_gen(gen_df, label=""):
    print(f"\nRunning simulation: {label}")
    sim = Simulator(
        template,
        gen_df,
        load_data,
        None,
        pd.to_datetime(start_date).date(),
        1,
        solver="gurobi",
        solver_options={},
        run_lmps=False,
        mipgap=RUC_MIPGAPS[grid],
        load_shed_penalty=1e4,
        reserve_shortfall_penalty=1e3,
        reserve_factor=0.05,
        output_detail=3,
        prescient_sced_forecasts=True,
        ruc_prescience_hour=0,
        ruc_execution_hour=16,
        ruc_every_hours=24,
        ruc_horizon=48,
        sced_horizon=SCED_HORIZONS[grid],
        lmp_shortfall_costs=False,
        enforce_sced_shutdown_ramprate=False,
        no_startup_shutdown_curves=False,
        init_ruc_file=None,
        verbosity=0,
        output_max_decimals=4,
        create_plots=False,
```

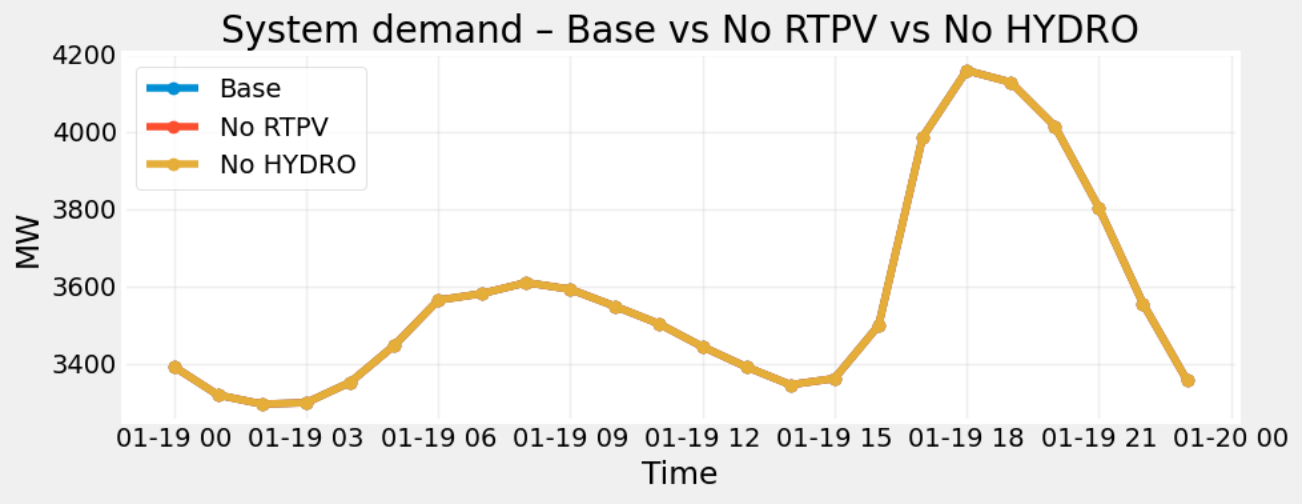
```
        renew_costs=None,
        save_to_csv=False,
        last_conditions_file=None,
    )
    return sim.simulate()

# All RTPV generators shut down
report_rtpv_off = run_sim_with_gen(gen_data_rtpv_off, label="No RTPV")

# All HYDRO generators shut down
report_hydro_off = run_sim_with_gen(gen_data_hydro_off, label="No HYDRO")
```

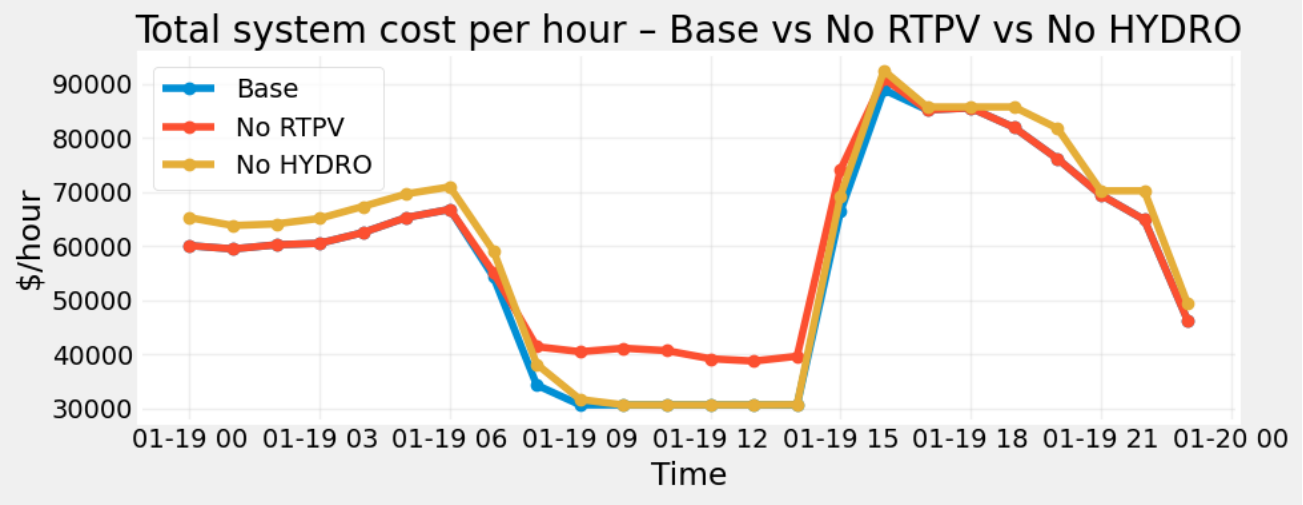
```
Running simulation: No RTPV

Running simulation: No HYDRO
```

In [39]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 3 scenarios
scenarios_raw_q22 = {
    "Base":     report_dfs,
    "No RTPV":  report_rtpv_off,
    "No HYDRO": report_hydro_off,
}

def prepare_hourly(rep):
    """
    Clean hourly_summary: add Datetime index and derived columns.
    Works whether Date/Hour are in the index or columns.
    """
    hs = rep["hourly_summary"].copy()
    hs = hs.reset_index()

    # timestamp
    hs["Datetime"] = pd.to_datetime(hs["Date"]) + pd.to_timedelta(hs["Hour"]

    # total cost
    hs["TotalCost"] = hs["FixedCosts"] + hs["VariableCosts"]

    # curtailment rate (% of available renewables)
    hs["CurtailmentPct"] = (
        hs["RenewablesCurtailment"] / hs["RenewablesAvailable"]
    ) * 100.0
    hs["CurtailmentPct"] = hs["CurtailmentPct"].replace([np.inf, -np.inf], r

    return hs.set_index("Datetime").sort_index()

scenario_hs_q22 = {name: prepare_hourly(rep) for name, rep in scenarios_raw_
```

In [40]:
```
def plot_metric_q22(metric, ylabel, title, scenarios=scenario_hs_q22):
    plt.figure(figsize=(10, 4))
    for name, hs in scenarios.items():
        if metric not in hs.columns:
            continue
        plt.plot(hs.index, hs[metric], marker="o", label=name)
    plt.xlabel("Time")
    plt.ylabel(ylabel)
    plt.title(title)
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.tight_layout()
```

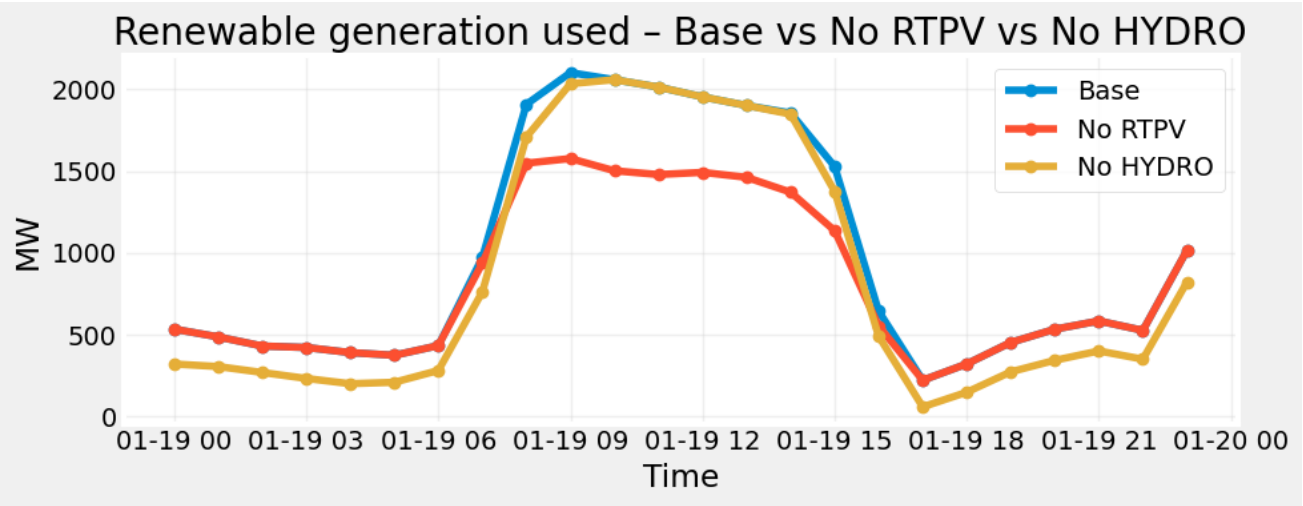
In [41]:
```
plot_metric_q22(
    metric="Demand",
    ylabel="MW",
    title="System demand — Base vs No RTPV vs No HYDRO",
)

# Total system cost per hour
plot_metric_q22(
    metric="TotalCost",
    ylabel="$/hour",
    title="Total system cost per hour — Base vs No RTPV vs No HYDRO",
)
```

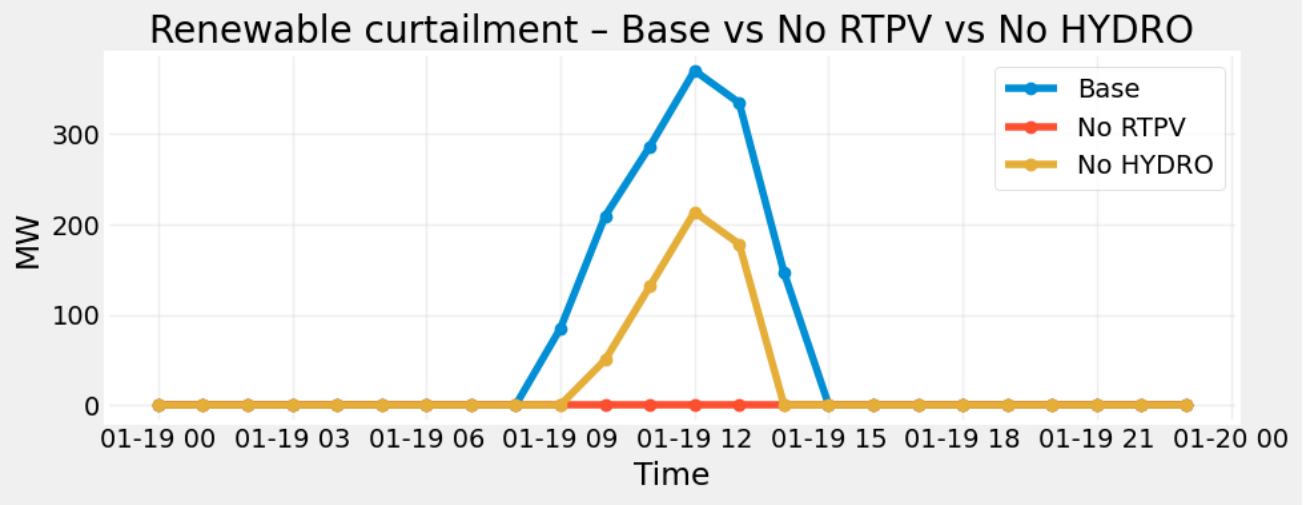




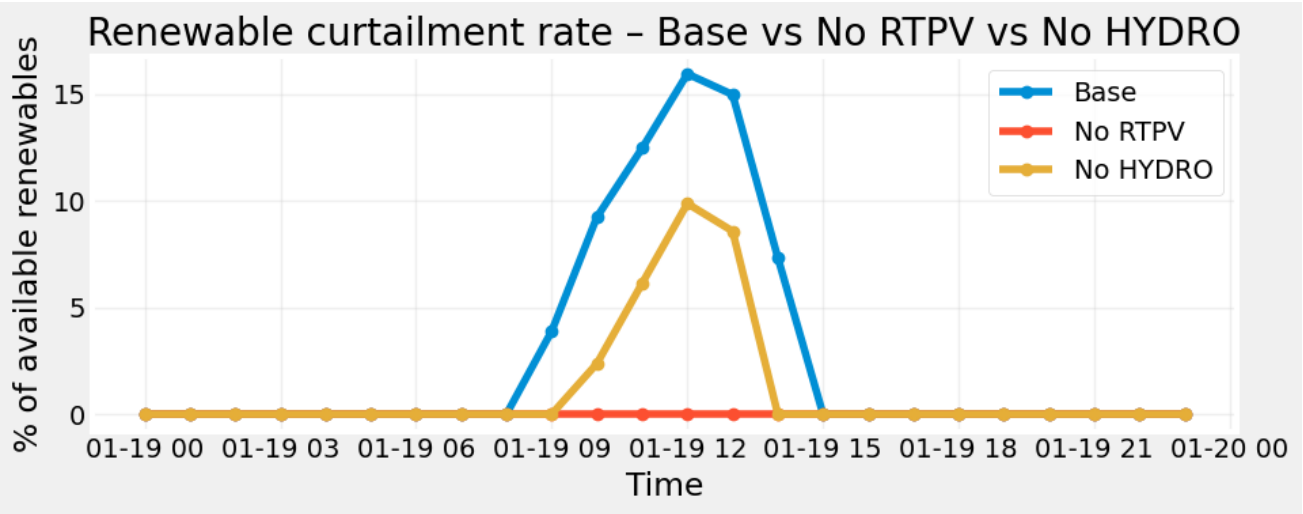
In [ ]:

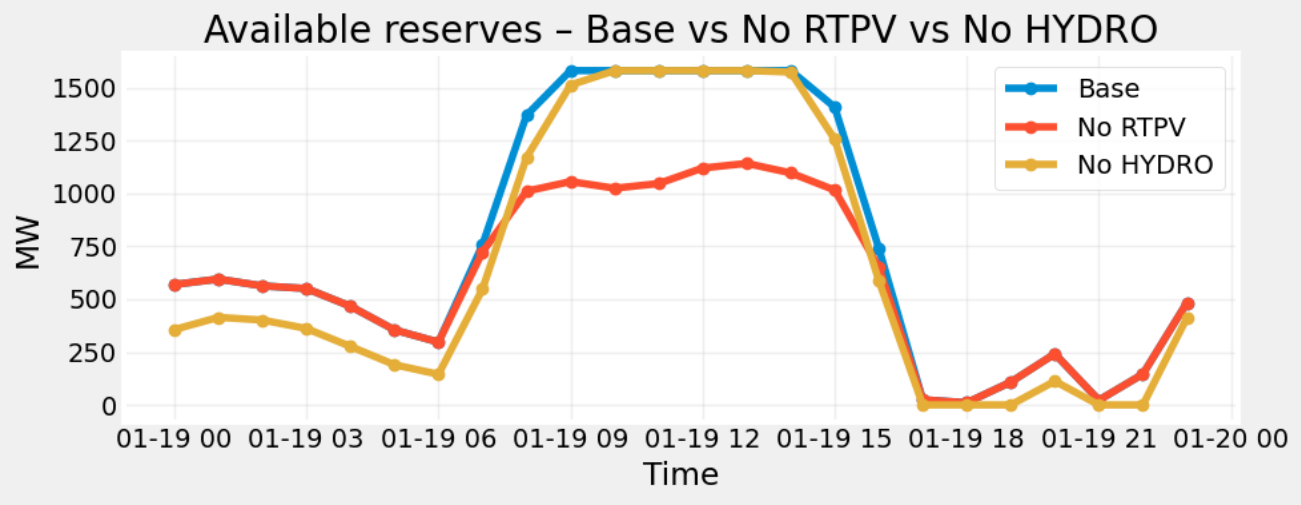
In [42]:
```
# Renewable generation used
plot_metric_q22(
    metric="RenewablesUsed",
    ylabel="MW",
    title="Renewable generation used — Base vs No RTPV vs No HYDRO",
)

# Renewable curtailment (MW)
plot_metric_q22(
    metric="RenewablesCurtailment",
    ylabel="MW",
    title="Renewable curtailment — Base vs No RTPV vs No HYDRO",
)
```

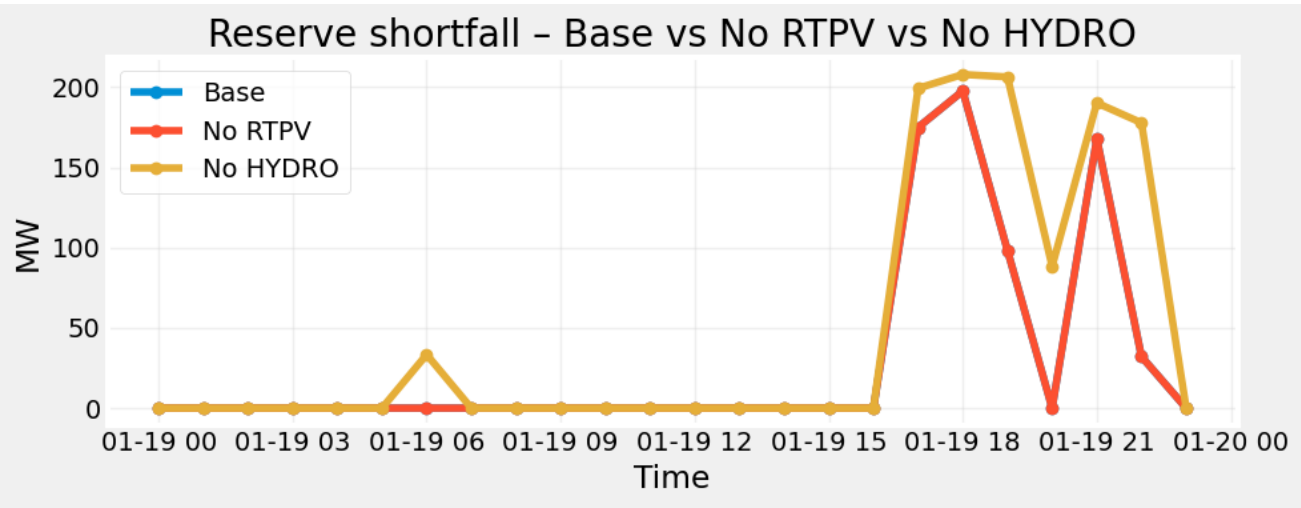




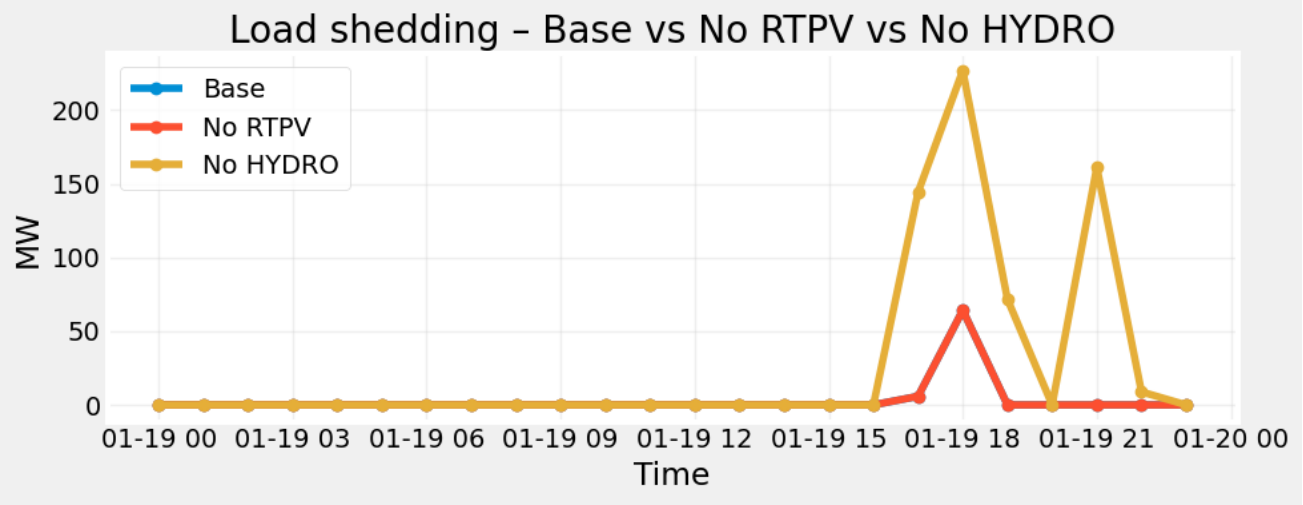
In [43]:
```
# Curtailment rate (%)
plot_metric_q22(
    metric="CurtailmentPct",
    ylabel="% of available renewables",
    title="Renewable curtailment rate — Base vs No RTPV vs No HYDRO",
)

# Reserves
plot_metric_q22(
    metric="AvailableReserves",
    ylabel="MW",
    title="Available reserves — Base vs No RTPV vs No HYDRO",
)
plot_metric_q22(
    metric="ReserveShortfall",
    ylabel="MW",
    title="Reserve shortfall — Base vs No RTPV vs No HYDRO",
)
```

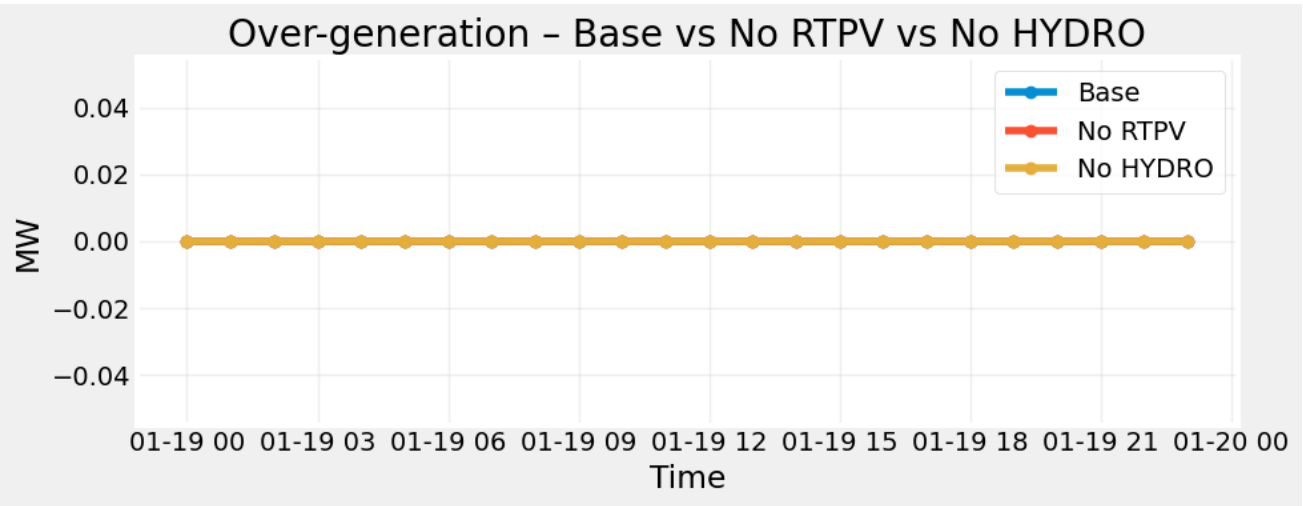


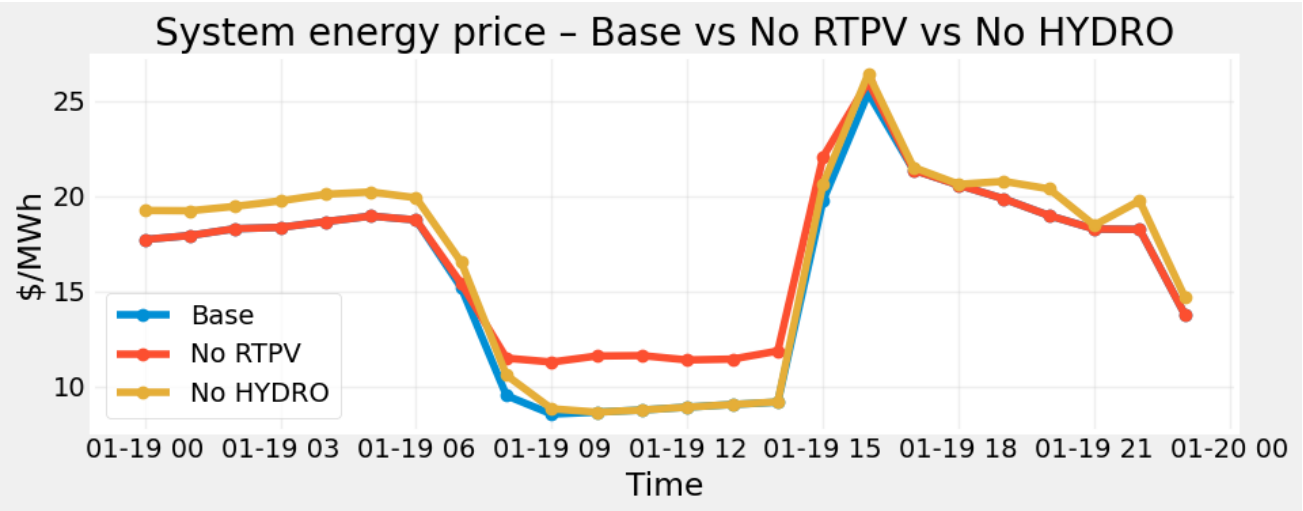




In [44]:
```
# Load shedding and over-generation
plot_metric_q22(
    metric="LoadShedding",
    ylabel="MW",
    title="Load shedding – Base vs No RTPV vs No HYDRO",
)
plot_metric_q22(
    metric="OverGeneration",
    ylabel="MW",
    title="Over-generation – Base vs No RTPV vs No HYDRO",
)

# Price
plot_metric_q22(
    metric="Price",
    ylabel="$/MWh",
    title="System energy price – Base vs No RTPV vs No HYDRO",
)
```

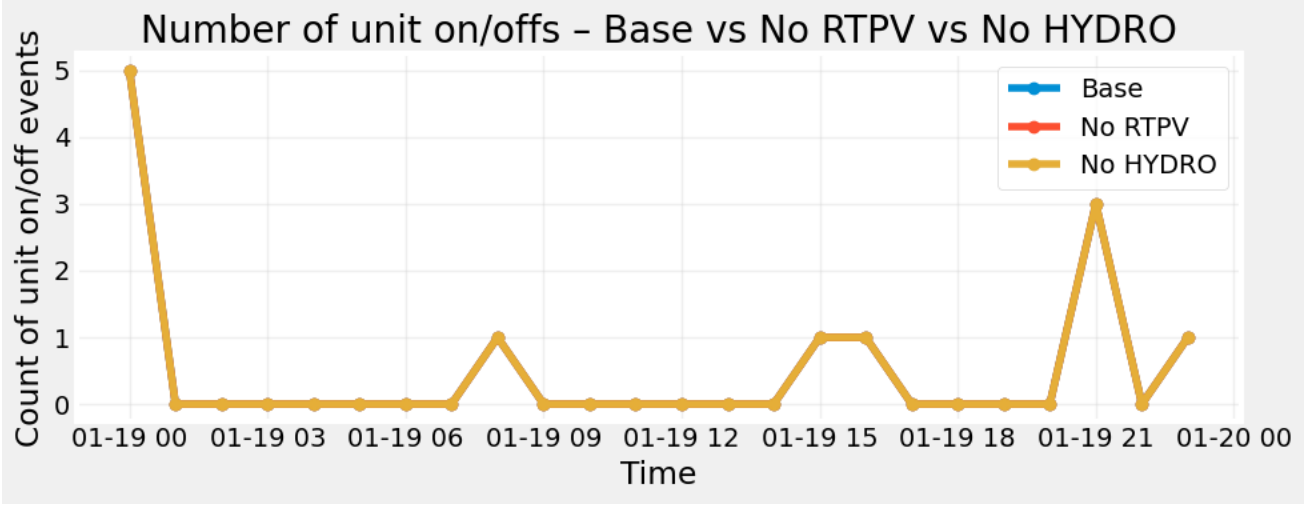


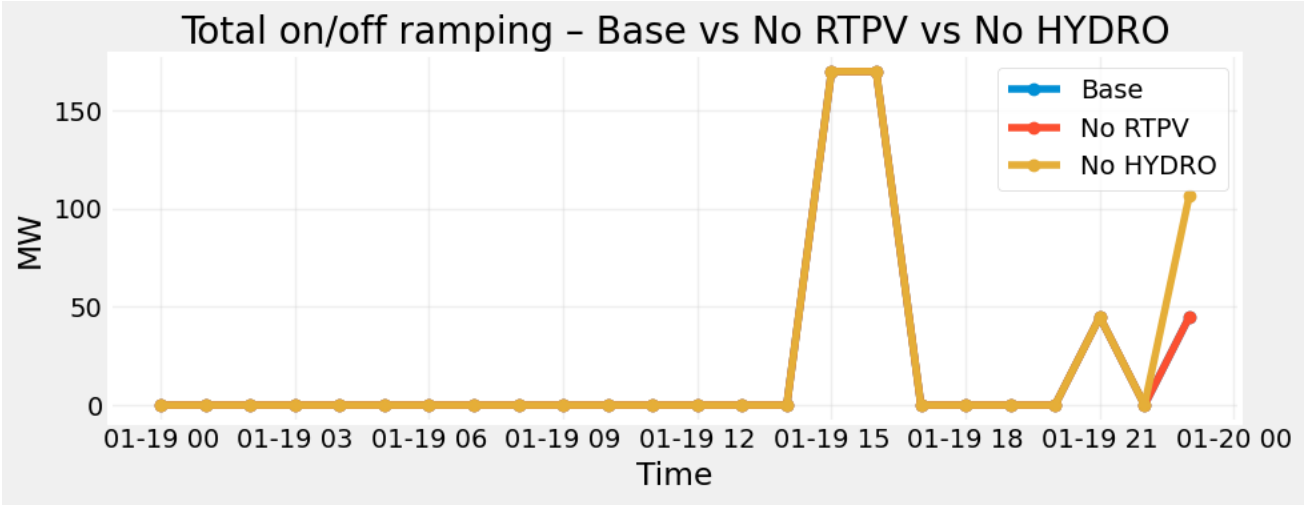


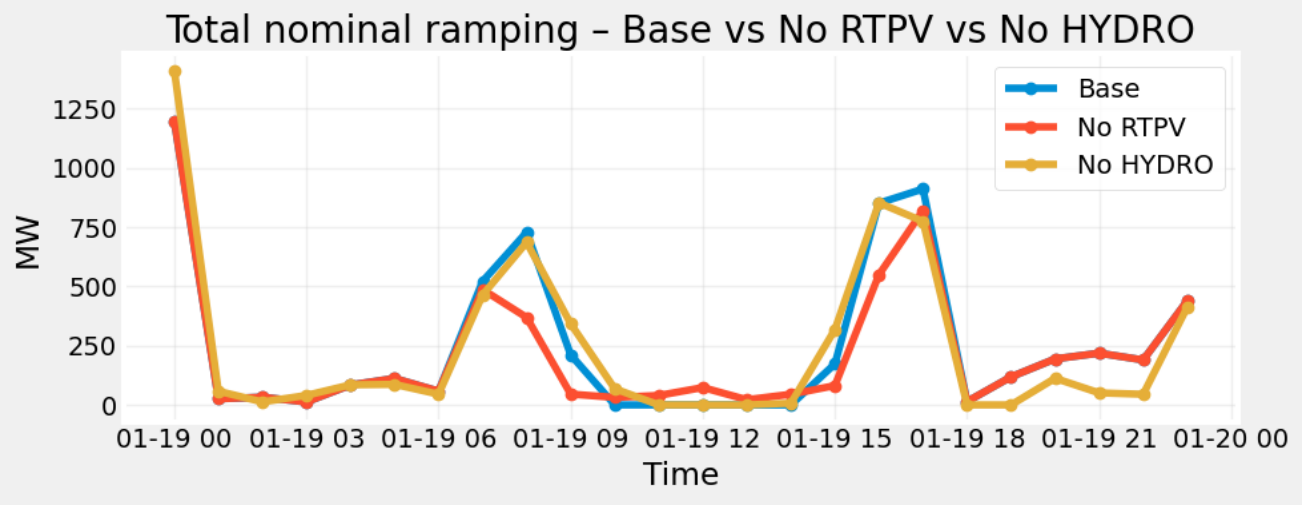


In [45]:
```
# Commitment & ramping behaviour
plot_metric_q22(
    metric="Number on/offs",
    ylabel="Count of unit on/off events",
    title="Number of unit on/offs — Base vs No RTPV vs No HYDRO",
)
plot_metric_q22(
```

```
    metric="Sum on/off ramps",
    ylabel="MW",
    title="Total on/off ramping — Base vs No RTPV vs No HYDRO",
)
plot_metric_q22(
    metric="Sum nominal ramps",
    ylabel="MW",
    title="Total nominal ramping — Base vs No RTPV vs No HYDRO",
)
```

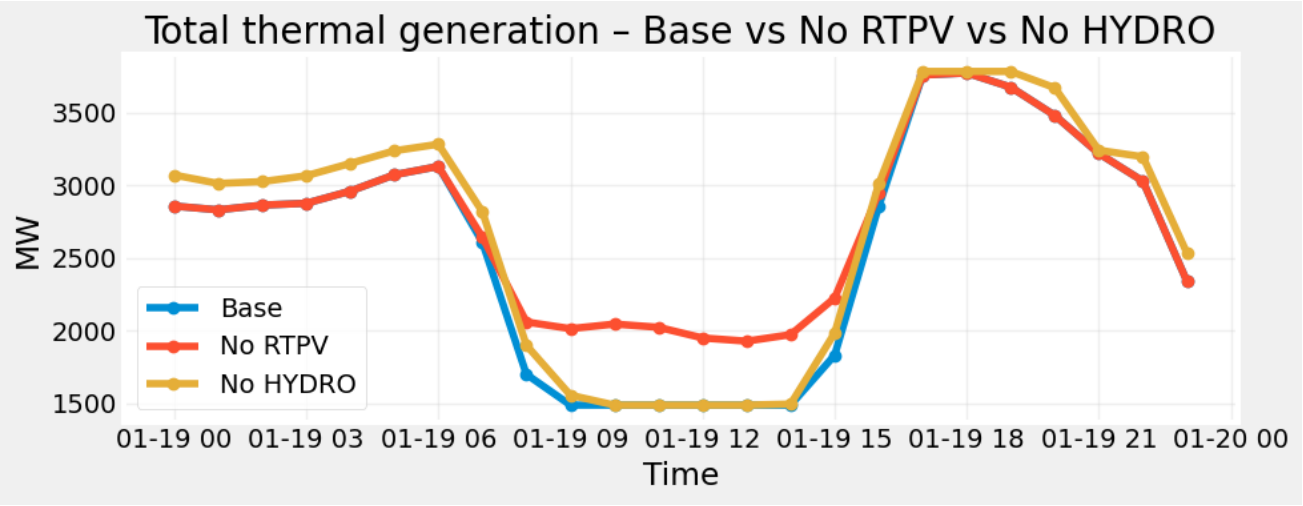


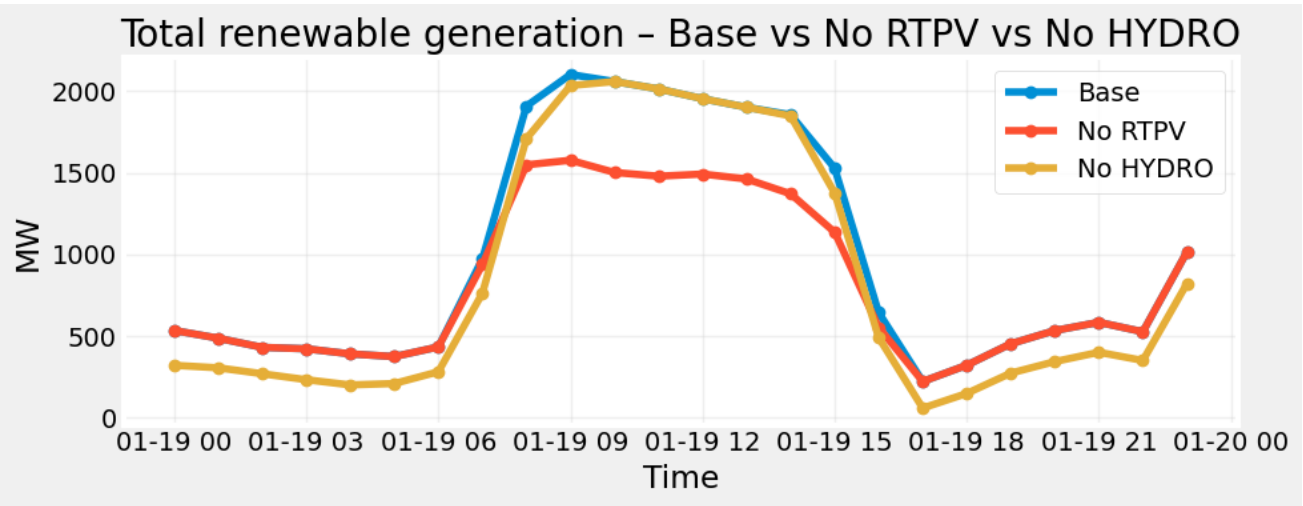




In [46]:
```
def thermal_dispatch_time_series(rep):
    td = rep["thermal_detail"].copy().reset_index()
    td["Datetime"] = pd.to_datetime(td["Date"]) + pd.to_timedelta(td["Hour"]
    return td.groupby("Datetime")["Dispatch"].sum().sort_index()

def renewable_dispatch_time_series(rep):
    rd = rep["renew_detail"].copy().reset_index()
    rd["Datetime"] = pd.to_datetime(rd["Date"]) + pd.to_timedelta(rd["Hour"]
    return rd.groupby("Datetime")["Output"].sum().sort_index()

thermal_ts_q22 = {
    name: thermal_dispatch_time_series(rep)
    for name, rep in scenarios_raw_q22.items()
}
renew_ts_q22 = {
    name: renewable_dispatch_time_series(rep)
    for name, rep in scenarios_raw_q22.items()
}

# Total thermal generation
plt.figure(figsize=(10, 4))
for name, series in thermal_ts_q22.items():
    plt.plot(series.index, series.values, marker="o", label=name)
plt.xlabel("Time")
plt.ylabel("MW")
plt.title("Total thermal generation – Base vs No RTPV vs No HYDRO")
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()

# Total renewable generation
plt.figure(figsize=(10, 4))
for name, series in renew_ts_q22.items():
    plt.plot(series.index, series.values, marker="o", label=name)
plt.xlabel("Time")
```

```
plt.ylabel("MW")
plt.title("Total renewable generation – Base vs No RTPV vs No HYDRO")
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
```





In [ ]:

In [ ]:

Main Observations:

- System demand remains identical across all cases, confirming that scenario differences arise solely from supply-side resource changes rather than demand variation.
- Removing rooftop PV (No RTPV) modestly increases total cost, especially during daytime hours, as the system loses low-cost distributed solar energy and must rely

more on thermal generation. Removing hydro (No HYDRO) causes the largest cost rise across all hours, particularly in the morning and evening/night, since hydro normally provides cheap, flexible generation to meet ramping and peak demands. Hydro's absence eliminates an essential balancing resource, resulting in persistently higher system costs.

- Both No RTPV and No HYDRO scenarios show reduced renewable usage. In the No RTPV case, daytime renewable output declines sharply, consistent with the loss of rooftop PV generation. In the No HYDRO case, renewable use declines throughout the entire day, not just midday, since hydro represents a steady renewable source that contributes during both solar and non-solar hours. Thus, RTPV primarily affects daytime renewable contribution, while hydro supports around-the-clock renewable availability.

- The Base and No HYDRO cases display notable midday curtailment due to PV oversupply, whereas the No RTPV case nearly eliminates curtailment by reducing total solar availability. This pattern confirms that PV (both utility and rooftop) is the primary driver of renewable excess during solar-rich hours, while hydro's removal has less impact on curtailment magnitude.

- The midday peak in renewable curtailment rate (~15%) persists in Base and No HYDRO (~10%) scenarios, indicating that removing hydro doesn't alleviate renewable oversupply. The near-zero curtailment in the No RTPV case again underscores that curtailment is driven by PV output levels rather than hydro flexibility.

- Reserves are highest in the No RTPV case before 6am and after 5pm, because the system compensates for missing PV with greater thermal commitment, leaving larger upward capacity margins. Conversely, reserves are lowest in the No HYDRO in these scenario, where the system loses hydro's ramping flexibility and headroom, particularly around morning and evening transitions. The Base system maintains a balanced reserve profile supported by both hydro and rooftop PV.

- Reserve shortfall occurs mainly during the evening ramp when solar output fades and load remains high. Both No RTPV and No HYDRO experience more frequent and larger shortfalls than the Base, but for different reasons: No RTPV faces higher evening net load (less PV to offset), while No HYDRO loses its flexible response capacity. The Base case has the smallest and shortest shortfalls, demonstrating its superior operational balance.

- Significant shedding occurs in the No HYDRO case during evening hours when both

demand and thermal ramping requirements peak. The No RTPV case also experiences occasional evening shedding, though less severe. None of the cases exhibit over-generation, confirming that renewable curtailment and proper dispatch avoided infeasible energy surpluses.

- Prices remain lowest and most stable in the Base case. Removing RTPV slightly raises prices across most hours during 8am to 2pm due to higher thermal generation costs. Removing hydro, however, triggers the steepest price increases, especially in the evening when hydro normally mitigates scarcity and ramps. This highlights hydro's central role in price stabilization and system cost efficiency, complementing RTPV's daytime cost reduction effect.

- The count of commitment changes is modest across all cases, with small clusters during morning start-ups and the evening ramp.

- The Base and No-Hydro cases show higher ramping during morning and evening transitions, indicating greater reliance on flexible generation when renewables fluctuate. The No-RTPV case exhibits slightly smoother ramps midday but sharper evening changes, highlighting rooftop PV's role in easing net-load variability during the day.

- Thermal generation rises most sharply in the No HYDRO case in early morning and post evvening, replacing the lost hydro energy and flexibility. In the No RTPV scenario, thermal generation also increases, but mainly during daylight and evening hours. The Base system maintains the lowest thermal output overall, benefiting from both distributed PV and hydro support. Renewable generation peaks around midday in all scenarios, but removing RTPV significantly reduces solar output between 08:00–17:00, lowering total renewable supply throughout the day. Eliminating hydro reduces renewable availability more uniformly—both during the night and during ramps—since hydro contributes across all hours. The Base case maintains the highest renewable utilization, demonstrating how RTPV provides daytime energy while hydro supports round-the-clock renewable supply.

In [ ]:

Overall:

- Comparing the Base, No RTPV, and No HYDRO scenarios reveals that rooftop PV (RTPV) and hydro serve highly complementary functions in maintaining a low-cost,

reliable, and flexible power system.

- The Base case provides the best balance: high renewable utilization, minimal curtailment, low costs, stable prices, and near-zero reliability issues.
- The No RTPV case eliminates a crucial daytime energy source, raising net demand, increasing thermal dispatch, and elevating costs—particularly during solar hours and evening ramps.
- The No HYDRO case poses a deeper operational challenge, removing the system's main source of ramping flexibility. This leads to more frequent shortfalls, load shedding, and volatile prices despite higher thermal generation.
- Together, the results confirm that RTPV primarily enhances daytime efficiency, while hydro ensures system reliability and ramping flexibility. Removing either resource increases dependence on thermal generation and erodes economic and reliability performance. The Base system's superior results underscore the synergistic importance of combining distributed solar with flexible hydro to achieve cost-effective, stable, and sustainable grid operations

In [ ]: